P1 1189181

REC'D 08 JUL 2004
WIPO PCT

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

July 01, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE.

APPLICATION NUMBER: *60/515,750*
FILING DATE: *October 30, 2003*
RELATED PCT APPLICATION NUMBER: *PCT/US04/09645*

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

M. Sias

M. SIAS
Certifying Officer

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

BEST AVAILABLE COPY

PROVISIONAL APPLICATION COVER SHEET [37 CFR 1.53(c)]

This is a request for filing a PROVISIONAL APPLICATION under 35 U.S.C. §111(b) and 37 CFR 1.51(a)(2)

Date : October 30, 2003
Docket No. : 51456/JDC/R268
EXPRESS MAIL NO. EV 351346083US

22582 U.S. PTO
60/515750
103003

Mail to: MAIL STOP PROVISIONAL PATENT APPLICATION

INVENTOR(S)/APPLICANT(S) (LAST NAME, FIRST NAME, MIDDLE INITIAL, RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY)
GERLA, Mario; Los Angeles, CA and SANADIDI, M. Yahya; Los Angeles, CA

_____ Additional inventors are being named on separately numbered sheets attached hereto.

TITLE OF THE INVENTION (280 characters max)
IMPLEMENTATION OF TCP WESTWOOD ABSE ON FREEBSD

APPLICANT(S) STATUS UNDER 37 CFR § 1.27

X Applicant(s) and any others associated with it/them under § 1.27(a) are a SMALL ENTITY

ENCLOSED APPLICATION PARTS

9 Specification (number of pages)
_____ Drawings (number of sheets)
_____ Assignment
_____ Other (specify):

FEE AND METHOD OF PAYMENT

X A check for the filing fee of $ 80.00 is enclosed.
The Commissioner is hereby authorized to charge any fees under 37 CFR 1.16 and 1.17 which may be required by this filing to Deposit Account No. 03-1728. Please show our docket number with any charge or credit to our Deposit Account. **A copy of this letter is enclosed.**
_____ No filing fee enclosed.

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

X No
_____ Yes, the name of the U.S. Government agency and the Government contract number are:

Please address all correspondence to **CHRISTIE, PARKER & HALE, LLP, P.O. Box 7068, Pasadena, CA 91109-7068, U.S.A.**

Respectfully submitted,

CHRISTIE, PARKER & HALE, LLP

By ________________
John D. Carpenter
Reg. No. 34,133
626/795-9900

PROVISIONAL APPLICATION FILING ONLY

Implementation of TCP Westwood ABSE on FreeBSD

UCLA Computer Science Department Network Research Lab

Express Mail No. EV351346083

1. Installation

We have a kernel source of TCP Westwood ABSE for FreeBSD Release 4.4.

1.1. Install FreeBSD 4.4

Don't forget to install the kernel source.

> Note: If there is *not* a /usr/src/sys directory on your system, then the kernel source has not been installed. The easiest way to do this is by running /stand/sysinstall as root, choosing Configure, then Distributions, then src, then sys. If you have an aversion to sysinstall and you have access to an "official" FreeBSD CDROM, then you can also install the source from the command line:

```
% mount /cdrom
% mkdir -p /usr/src/sys
% ln -s /usr/src/sys /sys
% cat /cdrom/sys/ssys.[a-d]* | tar -xzvf
```

1.2. Get and extract the Westwood ABSE archive

Get the archive of TCP Westwood ABSE kernel source.

Then, extract it.

```
% cd /usr/src
% tar zxvf westwood_ABSE-R45-121402.tgz
```

1.3. Compile and install Westwood kernel

```
% cd /usr/src/sys/i386/conf
% /usr/sbin/config WESTWOOD_ABSE
% cd ../../compile/WESTWOOD_ABSE
% make depend
% make
% make install
% reboot
```

1.4 If you are provided with TCPW Binary

If you are provided with the TCPW binaries the easiest way to install TCPW is to first save it on the hard drive. Then reboot FreeBSD and when it first begins to reboot press the ESC key and type the entire path to the kernel binary.

2. Configuration of Westwood Kernel

ABSE/BE kernel can be configured by running `sysctl'.

2.1. TCP version

We can select a TCP version among Westwood ABSE, Westwood BE and NewReno, and change it anytime. TCP version of each connection is determined at a connection's startup time.

(1) TCP ABSE
% sysctl net.inet.tcp.tcp_westwood=1
% sysctl net.inet.tcp.tcp_westwood_abse=1
(2) TCP BE
% sysctl net.inet.tcp.tcp_westwood=1
% sysctl net.inet.tcp.tcp_westwood_abse=0
(3) TCP NewReno
% sysctl net.inet.tcp.tcp_westwood=0

We can also enable TCP Astart or Agile Probing function in ABSE
% sysctl net.inet.tcp.tcp_westwood=1
% sysctl net.inet.tcp.tcp_westwood_abse=1
% sysctl net.inet.tcp.tcp_westwood_astart=1

Persistent Non-Congestion Detection (PNCD) is enabled by
% sysctl net.inet.tcp.tcp_westwood=1
% sysctl net.inet.tcp.tcp_westwood_abse=1
% sysctl net.inet.tcp.tcp_westwood_astart=1
% sysctl net.inet.tcp.tcp_westwood_ncd=1

3. Details of Implementation of TCPW ABSE Protocol

We have implemented TCPW ABSE protocol on Free BSD 4.4. In this section, we will present details of the implementation.

3.1. Issues

We have three main issues to consider in implementation of TCPW ABSE protocol. We will discuss the issues and our solutions here.

3.1.1. Cumulative ACK handling

TCPW ABSE Protocol estimated the bandwidth by a difference of ACK sequence number between two packets. We need to care the special situation like reordering and duplicate ACKs. Normally, we can not get the accurate information in such a situation.

3.1.1.1. Handling reordering packets

When a source node received a reordered packet which has smaller ACK sequence number than the previous packet, the sender can not predict how many bytes the destination node received in the corresponding time period. Thus, we neglect the reordered packets. The source maintains the maximum sequence number of ACK packets, and if the ACK sequence number of the received packet is less than the maximum one, it assumes that it is a reordering packet and just neglects it. If the maximum sequence number becomes too large due to some error, all packets are treated as reordering packets. To avoid this improper situation, the source node enforces that the maximum sequence number never exceeds the maximum sequence number of packets that have ever been sent. An example of Reordering is shown in Figure 1. Packets with ACK sequence number 600, 1100 and 2100 are neglected because their sequence numbers are smaller than the maximum sequence number.

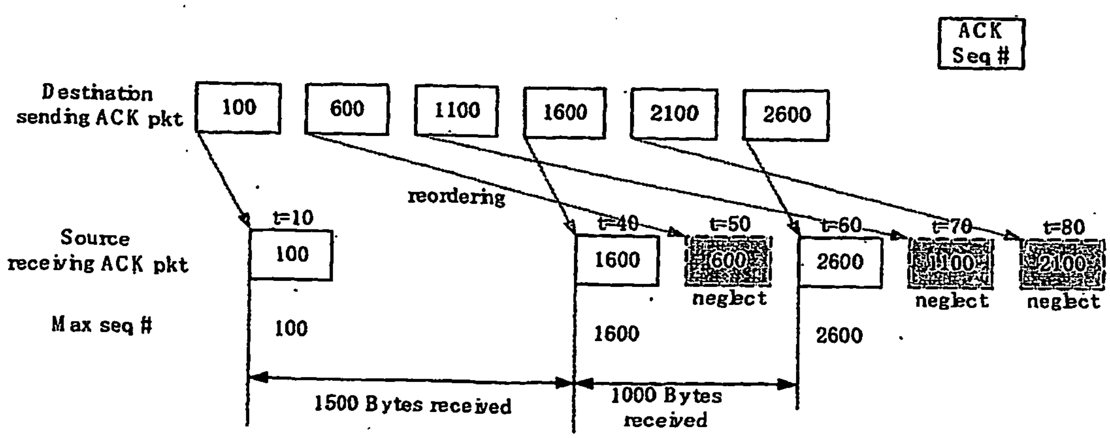

Figure 1 ACK packet Reordering

3.1.1.2. Expected ACK sequence number

TCPW ABSE Protocol estimates the bandwidth as a difference of ACK sequence number divided by a difference of receiving time between two packets. If the source receives a Duplicate ACK, we can not calculate the amount of bytes received. To solve this problem, we introduce the expected ACK sequence number instead of the real ACK sequence number. The source assumes that a duplicate ACK packet corresponds to an average TCP segment size. The expected sequence number is increased by the average TCP segment size. The average size is calculated as exponential averaging of sending packets. Like calculation of the maximum ACK sequence number, it enforces that the expecting sequence number never exceeds the maximum sequence number of the packet that have ever been sent. Examples of the expected sequence number calculation are shown in Figure 2 and Figure 3.

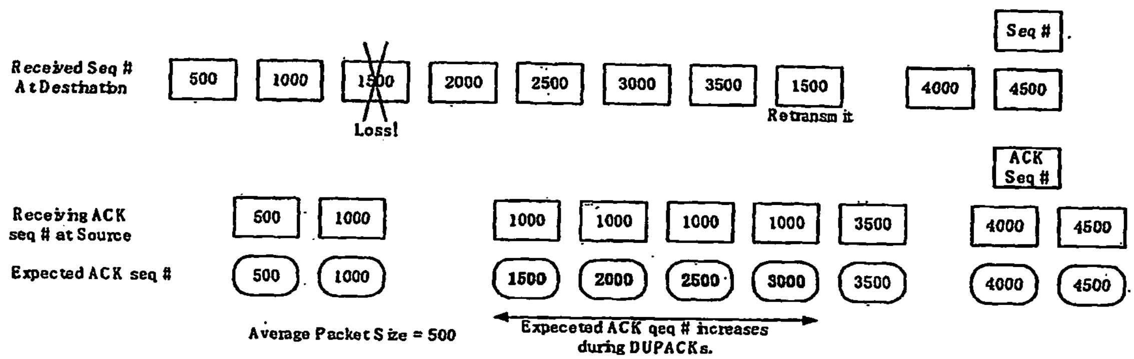

Figure 2 Example of the expected ACK seq % calculation in duplicate ACKs.

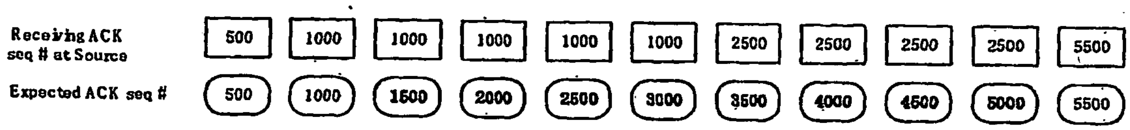

Figure 3 Example of the expected ACK seq % calculation in partial ACKs

3.1.2. CPU clock granularity problem

The CPU clock in FreeBSD is called ticks. The default unit of ticks is 10msec. This low granularity causes two major problems.

3.1.2.1. Several packets can be received at the same time unit.

When several packets are received at the same unit, we can not estimate the bandwidth between these packets because the time interval is 0sec. We only use the first received packet in each ticks to calculate bandwidth. Figure 4 shows an example of receiving several packets in one ticks.

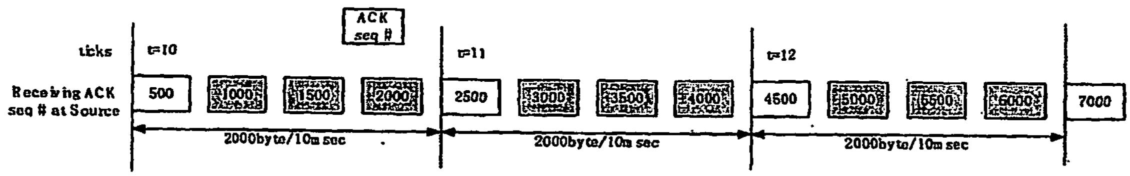

Figure 4 Receiving several packets in one ticks.

We have two benefits by using one packet for each 10msec. At first, we only need bandwidth calculation once per 10msec. It reduces the CPU load in high speed connections. The other is that we only need to accommodate one ACK packet information per 10msec. It reduces an amount of the buffer for storing ACK packet information. However, we must care effects of neglecting the packets in the same tick. We will evaluate difference of simulation and this implement in the experiment section.

3.1.3. ACK History Management

TCPW ABSE Protocol needs to keep a number of ACK information, sequence number and receiving time, to calculate the Bandwidth sampling. The maximum time interval for storing ACK information is determined by the sampling interval T. The maximum value of T is one RTT. Thus, the number of packets needs to be stored: N is calculated as

$$N = \frac{MaximumWindowSize}{PacketSize}.$$

If we assume that the Maximum Windows Size is 64kbytes and the packet size is 512 bytes, we need 128 packets. However, if we want to achieve the high throughput in a large RTT like 45Mbps in RTT=70ms, the windows size should be more than 3Mbytes, and N should be more than 6kbytes.

Fortunately, due to 10ms ticks restriction, we only accommodate one packet for each ticks unit. Thus, we can calculate N as

$$N = \frac{RTT}{ticks\_time\_unit(default\_10m\sec)}$$

Now, we can determine N indifferent to throughput. If we set N=256, we can support TCP connections with less than 2.5sec RTT. Though 2.5sec RTT is enough large in the current Internet, we also need to handle a connection with more than 3sec RTT. To support a very long RTT, we have thought upon the three ideas.

(1) Assign N as a kernel parameter. An administrator can change N by 'sysctl' if he needs.

(2) Increase/decrease N in accordance with the RTT value dynamically. It must make the code complex.

(3) Introduce a new parameter, the minimum time interval for storing the ACK information, and change it adaptively based on RTT so that we can accommodate all packets within T without changing N.

We select (3) in the current implementation because we can determine N indifferent to RTT value.

4. The code Overview

Bandwidth Estimation Function is called whenever an ACK packet is received. The flow chart of Bandwidth Estimation function is shown in Figure 5.

(1) Reordering

If the packet is reordered, the calculation is cancelled.

(2) Updating Expected sequence number

(3) Check receiving time

So, if the packet is received at the same tick as the previous stored packet, the calculation is cancelled.

(4) Stored to the ACK information Array

The expected sequence number and receiving time is stored in the ACK information array.

(5) Calculate Filter Gain Adaptation.

Instability U_k and ABSE filter parameter τ_k is calculated as

$$U_k = \beta \cdot U_{k-1} + (1-\beta) \cdot |s_k - s_{k-1}|$$

$$U_{max} = \max(U_{k-10} \ldots U_k)$$

$$\tau_k = RTT + N \cdot RTT \frac{U_k}{U_{max}}$$

U_{max} is calculated as the maximum instability in the ten most recent observations. U_{max} is possibly becomes 0, and it causes 0 division error. We define τ_k=RTT if U_{max}=0.

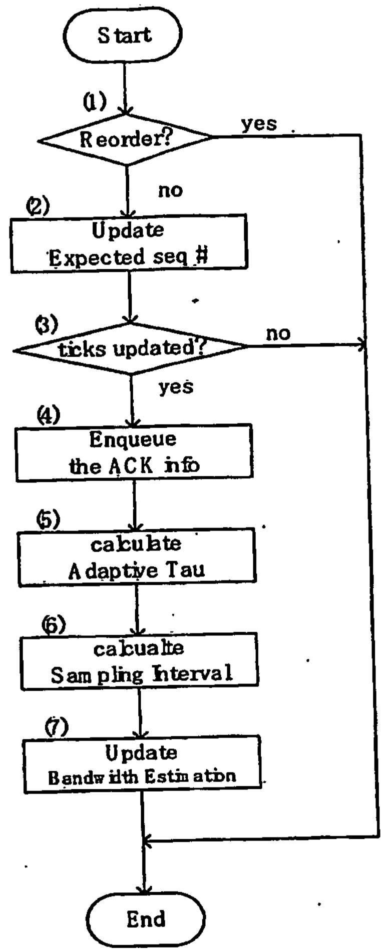

Figure 5 Flow chart of Bandwidth Estimation functions

(6) Calculate ABSE adaptive sampling interval

When a sender received a packet, the bandwidth sampling is computed based on the amount of data acknowledged during the interval T. T is calculated as

$$T_k = RTT \cdot \frac{cwin - C \cdot RTT_{min}}{cwin}$$

(7) Update Bandwidth Estimation

From the ACK information array, get the oldest and newest ACK during the interval T, and calculate the sample bandwidth s_k and the estimated bandwidth $\hat{s}_k$. If T is less than or equal to 0 or there are only one ACK during the time interval T, it uses the two

least stored ACK packets.